Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 10 (2013) 715 – 723

Procedia Technology

International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) 2013

Plagiarism Detection by Identifying the Equations

Debotosh Bhattacharjee [a] and Sandipan Dutta[b]

[a] *Department of Computer Science and Engineering, Jadavpur University, Kolkata*
[b] *Department of Information Technology, Heritage Institute of Technology, Kolkata*

Abstract

In academia Plagiarism means copying of other work without author's permission. Presently available system mainly focuses on software plagiarism. They mainly based on token analysis, linguistic patterns, taxonomy and textual features. In this paper we mainly concentrate on research papers to check whether the documents are plagiarized or not. So far not much work has been done to detect plagiarism in research document. Our work focuses on the similarity of different simple equations present in a document. It can easily extract those equations from the documents, compare them even if the variables are changed in plagiarized document with the original one and can detect if the document is plagiarized or not. This method will not work if the research paper does not contain any equation.

*Keywords:*Plagiarism;Equation;Whitespace;Scan;

1. Introduction

We believe plagiarism is a serious threat in case of research work. Detection of plagiarism can be either manual or computer-assisted. Manual detection requires substantial effort and excellent memory, and is impractical in cases where too many documents must be compared, or original documents are not available for comparison. Computer-assisted detection allows vast collections of documents to be compared to each other, making successful detection much more likely. Presently available software mainly focuses on 'word by word' comparison. But definition of plagiarism is not restricted to copy the document but also copying the idea in different format. For a current view of literature survey [1] we noticed an approach to detect Spatial Plagiarism Similarity and Temporal Plagiarism

E-mail address:meet2sandipan@gmail.com

doi:10.1016/j.protcy.2013.12.414

Similarity and then developing an Evolutionary Plagiarism Probability model. Another method is the modification [3] of CC Finder [4] which is a famous tool in token based analysis. But maximum approaches are limited to detect software plagiarism. Only [2] has proposed some technique by analyzing Lexical features, Syntactic features, Semantic features, Structural features of the document to detect plagiarism in document. In this paper, we will present a technique which will extract the equation present in the target document and then comparing each equation with multiple source document's equations will conclude if the target document is copied from the source or not. Even if the variable name is different it will work successfully.

2. Proposed Methodology

The methodology can be summarized below.

2.1. Database Creation:

Our survey indicates that there is some specific font for the reputed journals. Number of variation for the font is restricted to two or three type's viz. Times New Roman. So we first develop the database of different character of predefined font. The database consists of A-Z, a-z, 0-9, operators like +, -, *, / and some symbol like ∑, √, ±, π, θ, λ, ∏, (,), θ,', ','; All the character of the database are collected by analyzing different research document. Each character is created manually and stored in the database in image format. All we tried to convert a particular character into that type which similar to the type of research papers. By doing this actually we are making the comparison process easier.

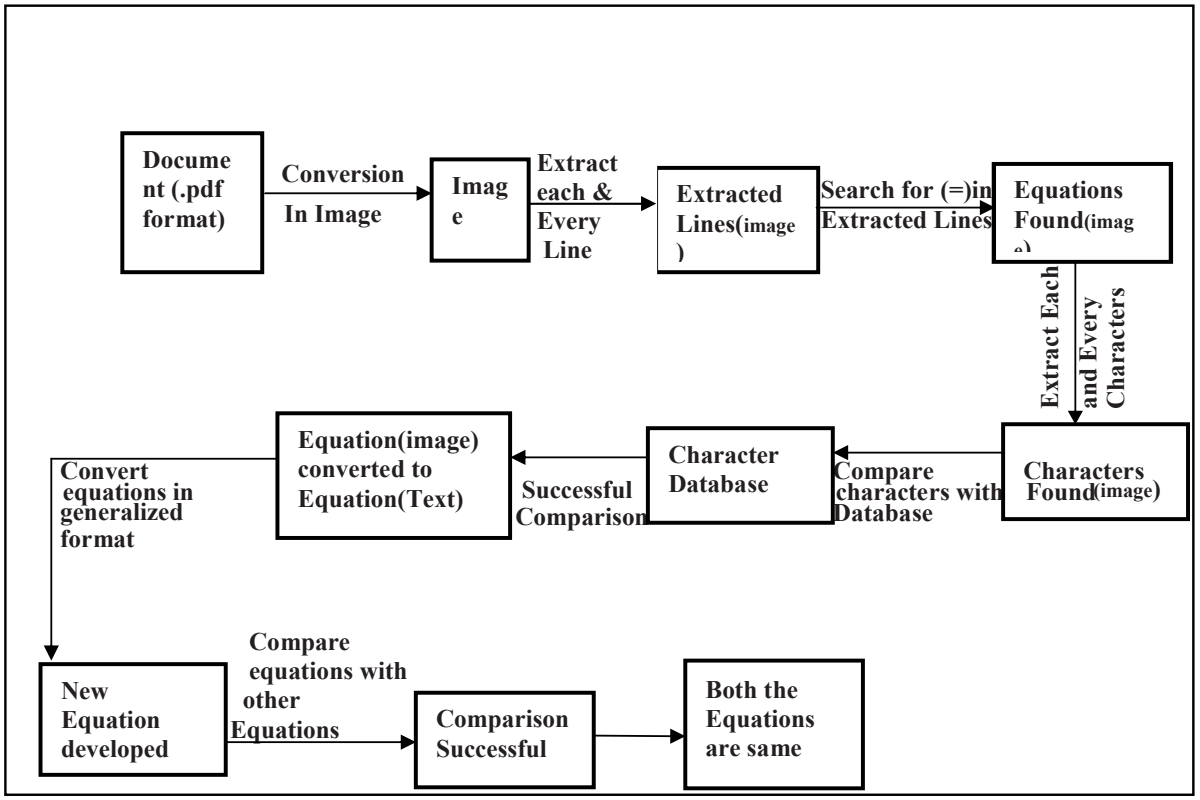


Fig1: Block Diagram of the System

Sample database table is shown below

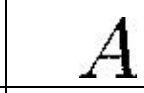
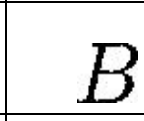
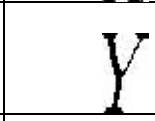
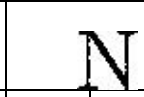
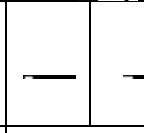
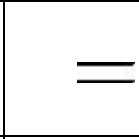
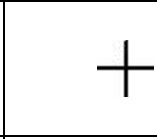
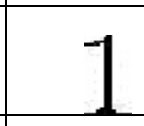
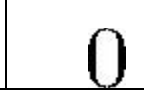
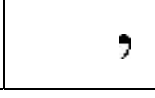
A			X			A	
B			Y			B	
N			Z			C	
-			(			[	
=			+			Θ	
1			**2**			**7**	
0			,			;	

Fig 2: Sample Database Table

From the sample table, shown Fig 2 it is very clear that characters are of different sizes. So, we need to make it of same sizes for smooth operation.

2.2. Preprocessing on target document:

Initially document is in .pdf format. So we first need to convert the entire document in image (.jpeg) format. That has been done by Scanning. Then our next job is to extract each and every line so that in the next step we can extract each and every character from the corresponding line. Once we extract each and every character then it is very easy to search and 'equal to' (=) sign which means that particular line is an equation.

*2.2.1. Scan***:**

As all the research papers are available in pdf format so we first convert the entire document into image format by scanning the entire document in 300dpi. The correctness of the output strongly depends on the proper scanning of the document. Portion of a scanned page is shown in Fig 3.

Proof. Suppose

$$AX_1 = \lambda_1 X_1,\ AX_2 = \lambda_2 X_2, \tag{7.4}$$

where X_1 and X_2 are non–zero column vectors, $A^t = A$ and $\lambda_1 \neq \lambda_2$.

We have to prove that $X_1^t X_2 = 0$. From equation 7.4,

$$X_2^t A X_1 = \lambda_1 X_2^t X_1 \tag{7.5}$$

and

$$X_1^t A X_2 = \lambda_2 X_1^t X_2. \tag{7.6}$$

From equation 7.5, taking transposes,

$$(X_2^t A X_1)^t = (\lambda_1 X_2^t X_1)^t$$

so

$$X_1^t A^t X_2 = \lambda_1 X_1^t X_2.$$

Hence

$$X_1^t A X_2 = \lambda_1 X_1^t X_2. \tag{7.7}$$

Finally, subtracting equation 7.6 from equation 7.7, we have

$$(\lambda_1 - \lambda_2) X_1^t X_2 = 0$$

and hence, since $\lambda_1 \neq \lambda_2$,

$$X_1^t X_2 = 0.$$

THEOREM 7.1.5 Let A be a real 2×2 symmetric matrix with distinct eigenvalues λ_1 and λ_2. Then a proper orthogonal 2×2 matrix P exists such that

$$P^t A P = \text{diag}\,(\lambda_1,\ \lambda_2).$$

Fig: 3

2.2.2. Extracting lines:

After scanning extract every line present in the document. For this go for horizontal scanning of the entire image. Then take whitespaces between two lines as a separator. If in a row the number of white pixels is equal to the total number of columns in the image, then conclude that this is a white space separator. Now, next objective is to locate consecutive lines of white pixels. Collection of white pixels in above manner termed as white pixel block. In this way, if two consecutive white pixel blocks has been located, then conclude that a black pixel line is present in between two white pixels blocks. By following this technique extract lines from different images.

Algorithm 1 extracts every line and the sample output is shown below.

Algorithm 1:

Input: Document in image format.
Output: All the lines are separated.

Step 1: Calculate the size (X,Y) of the image.
Step 2: initialize i=1 and j=1;
Step 3: while (i<=x)
Step 4: while (j<=y)
Step 5: locate the first black pixel (say P,S).
Step 6: set k=p+2;
Step 7: start scanning from (k,1).
Step 8: calculate the number of white pixel (say c)
Step 9: if(c=Y) then
Step 10: p1=k-1;
Step 11: end if
Step 12: Extract the block between p and p1.
Step 13: Set p=p1+1;
Step 14: go to Step 4.
Step 15: End while
Step 16: End while.

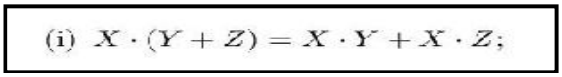


Fig: 4 Output of Algorithm 1

2.2.3 Extracting every character:

After extracting the lines next job is to extract each and every character from the extracted line by using the Algorithm 2. Here we go for vertical scanning. To extract individual character we also take the help of whitespace separator. Like the previous one if in a column the number of white pixels is equal to the total number of rows in the image, then conclude that this is a white pixel separator. Then search for white pixel block. In this way if two consecutive white pixel blocks has been located, then it can be concluded that a character is present between two white pixels blocks. In this way characters can be easily extracted from different images.

Algorithm 2 will extract every line and the sample output is shown below.

Algorithm 2:

Input: Line containing equation.
Output: All the characters in the line are separated.

Step 1: Rotate the line 90degree clockwise.
Step 2: Calculate the size (X,Y) of the image.
Step 3: initialize i=1 and j=1;
Step 4: while (i<=x)
Step 5: while (j<=y)
Step 6: locate the first black pixel (say P,S).
Step 7: set k=p+2;
Step 8: start scanning from (k,1).
Step 9: calculate the no of white pixel (say c)
Step 10: if(c=Y) then
Step 11: p1=k-1;
Step 12: end if
Step 13: Extract the block between p and p1.
Step 14: Set p=p1+1;
Step 15: go to Step 4.
Step 16: End while
Step 17: End while.

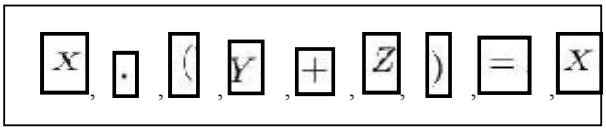


Fig 5: Output of Algorithm 2 for input given in Fig 4

By the two processes mentioned above we can find all the lines and all the characters present in the document. Now, we will find those line which contains an equality (=) sign. Because an equality (=) in a line signifies that line contains an equation. Equation can be identified by the following Algorithm 3.

Algorithm 3:

Input: Individual character.
Output: Detected whether it is a equation(=) or not.

Step 1: Calculate the size of the image(x,y)
Step 2: initialize i=1,j=1.
Step 3: while(i<x)
Step 4: while(j<y)
Step 5: Locate the first black pixel. (p,s).
Step 6: set s1=s+2;
Step 7: Locate the white pixel for same value of p(
say(a,b)
Step 8: set t=b-1;
Step 9: set z=p+2.
Step 9: Locate the first black pixel for same value of
z say(p1,s2).
Step 10:set s3=s2+2.

Step 11: Locate the white pixel for same value of p
say(a,b)
Step 12: set t1=b-1;
Step 13: if ((s=s2) and (t=t1) and ((p1-p)>2) then
conclude that this is an equation(=).
Step 14: End while
Step 15: End While

3. Matching Methodology

Once we find those lines which contains an equation, next objective is to convert those equations into a textual format from image format. The reason behind this conversion is to compare each character. To convert the equations into textual format compare the extracted image of character of the equation with the stored image in the database. Before comparison remove extra white spaces around the character. Once this comparison is successful create a text file and write the corresponding equation into that text file. Once these types of text files containing equations are created compare those equations and derive the result.

3.1. White Space removal:

The character may contain unnecessary white spaces around itself. So, the next step before resizing the image is to remove these extra white pixels to improvise the comparison process of character with database. To remove whitespace will use following Algorithm 4.

Algorithm 4:

Input: Character with white spaces.
Output: Character without white spaces.

Step 1: Calculate the size of the image(X,Y)
Step 2: set i=1,j=1.
Step 3: while(i<Y)
while(j<X)
If (Image (i,j)=0)
Store i values in P1 array.
Store j values in P2 array.
End If
End while.
End while.
Step 3: Identify maximum and minimum value from
P1 array say(s, s1).
Step 4: Identify maximum and minimum value from
P1 array say (t, t1).
Step 5: Extract the segment from (s,t) and (s1,t1).
Step 6: End while.
Step 7: End while.

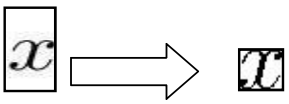


Fig 6: Output of Algorithm 4

3.2. Comparison with the database:

Once resized image is obtained compare it with the stored database image and can conclude the character from the image. That means conversion of equation from image to text is ready.

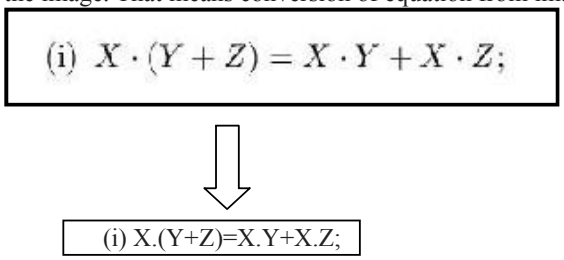


Fig 7:

Table I: Success result for identification of Equation

Document Name	No of Equation Actually Present	No Of Equation Identified	Success (%)
1.jpeg	4	3	75
2.jpeg	10	8	80
3.jpeg	2	2	100
4.jpeg	15	12	80
5.jpeg	8	8	100
6.jpeg	13	12	92
7.jpeg	10	10	100
8.jpeg	9	9	100
9.jpeg	19	17	89.47
10.jpeg	9	8	89

3.3. Comparison between the original and suspected document:

For successful comparison of different equation create different documents. Put some equations in different documents and in different format i.e. the meaning of the equation is same but variable names are different. Then by the above described process all the equation is identified from different documents and converted into textual format. Next job is to compare every equation between all the documents. For that first of all convert all the equation into a generalized format.
For example, consider the following equation:

a+b+c+d=0 (1)

The generalized format of this equation is

v+v+v+v=0 (2)

i.e. replace every variable with a predefined variable v.

Again consider another equation

c+d+a+b=0 (3)

Now, both the equation is same but their variable name is different. Replace it by a predefined variable v and it will be v+v+v+v=0 (4)

Then convert it into postfix format. In this case both the postfix will be same

vv+v+v+=0 (5)

Then by comparing both the equation it can be concluded that the equation is same. Once it is identified that a threshold number of equation is identical with another document then conclude that the document is plagiarized.

Table 2: Sample Result Analysis of identical document

Document Name	Equation Identified	Document Name	Equation Identified	Remarks
1.jpeg	a+b+c+d	2.jpeg	c+d+b+a	Identical
3.jpeg	a+b+c+d	4.jpeg	(a+b)+(c+d)	Identical
5.jpeg	a+b+c+d	6.jpeg	(c+d)+(b+a)	Identical
7.jpeg	(a+b)*(c+d)	8.jpeg	(e+f)*(p+q)	Identical

4. Conclusion:

There is some drawback of this methodology. It will not work efficiently if the document is not properly scanned. Future effort will be to make the process scan independent. Again this process will work if a sufficient number of equations present in the document. But future effort will be for all type of document whether that is equation based or not. Also this process is restricted to research paper only but future effort will be to develop a system which can operate efficiently on any type of document.

References

1. Detecting and Tracing Plagiarized Documents by Reconstruction Plagiarism-Evolution Tree by Chang-Keon Ryu, Hyong-Jun Kim, Seung-Hyun Ji, Gyun Woo, and Hwan-Gue Cho

2. Understanding Plagiarism Linguistic Patterns, Textual Features and Detection Methods by Salha M. Alzahrani, Naomie Salim, and Ajith Abraham in IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, Vol. 42, No. 2, March 2012

3.Type Redefinition Plagiarism Detection of Token-Based Comparison by Lifang Han1, Baojiang Cui,Ru Zhang, Zhongxian Li, Jianxin Wang, Yongle Hao in 2010 International Conference on Multimedia Information Networking and Security

4. Toshihiro Kamiya,Shinji Kusumoto, and Katsuro Inoue. CCFinder: A multilinguistic token-based code clone detection system for large scale source code. IEEE Transactions on Software Engineering, 28(7):645–670, July 2002.

5. R. M. Howard Plagiarism: Some sources on attitudes, definitions, and detection methods, 2007

6. L. Guterman "Copycat articles seem rife in science journals, a digital sleuth finds", Chronicle of Higher Education, 2008 [online] Available: http:// chronicle.com/daily/2008/01/1362n.htm